次の問9から問13までの5問については，この中から1問を選択し，選択した問題については，答案用紙の選択欄の(選)をマークして解答してください。

なお，2問以上選択した場合には，はじめの1問について採点します。

問9　次のCプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

5台のバスが走行する路線上の10か所の停留所（以下，バス停という）のそれぞれに取り付けられた表示装置に，バスの運行時間中，次のバスが到着するまでの待ち時間（以下，到着待ち時間という）を分単位で表示するシステムがある。関数update_wait_timeは，このシステムにおいて，各バス停に表示する到着待ち時間を計算するプログラムである。表示する到着待ち時間は，関数update_wait_timeが実行されたときに更新される。

(1)　関数update_wait_timeの引数は，次のとおりである。ここで，引数の値に誤りはないものとする。

bus_id	バスの車体番号
busstop	バス停の番号
route	路線表（構造体BUSSTOPの配列）
bus	路線上を走行するバスの一覧（構造体BUSの配列）

(2)　構造体BUSSTOPの構造は，次のとおりである。

```
typedef struct {
    int  std_time;    /* 前のバス停からこのバス停までの
                         標準所要時間（分） */
    int  wait_time;   /* 到着待ち時間（分） */
} BUSSTOP;
```

(3)　構造体BUSの構造は，次のとおりである。

```
typedef struct {
    int  id;          /* バスの車体番号 */
    int  cur_pos;     /* バスの走行位置 */
} BUS;
```

(4) 路線表 route は，バス停の番号（以下，バス停番号という）0～9 を添字とする配列である。route には，図 1 のようにデータを格納する。

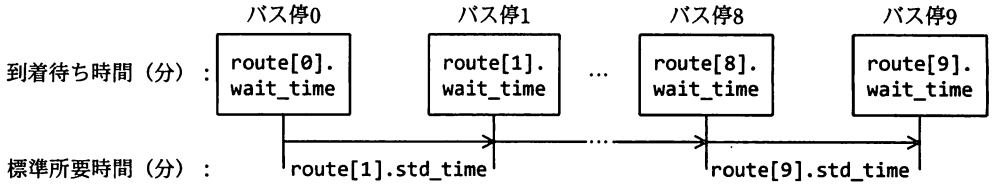

図 1　路線表 route のデータ

① route[i].std_time（i=1, 2, …, 9）は，前のバス停（i−1）からこのバス停 i までの標準所要時間である。バス停 0 における route[0].std_time の値は 0 とする。

② 各バス停 i の到着待ち時間 route[i].wait_time（i=1, 2, …, 9）は，次に到着するバスが最後に発車したバス停 a からこのバス停 i までの標準所要時間 route[k].std_time（k=a+1, …, i）の合計である。乗客の乗降にかかる時間は 0 とする。バス停 0 における route[0].wait_time の値は 0 とする。次に到着するバスがバス停 0 を発車していないとき，そのバス停 i における到着待ち時間 route[i].wait_time には 0 を格納する。

(5) バスの走行位置 bus[j].cur_pos（j=0, 1, …, 4）には，走行中の区間番号を格納する。あるバス停から次のバス停までの区間に対する区間番号には，区間の始まりとなるバス停番号を割り当てる。バス停 0 を発車していないバスの走行位置 bus[j].cur_pos には −1 を格納する。また，バス停 9 に到着したバスの走行位置 bus[j].cur_pos には 9 を格納する。

(6) プログラム中で使われる変数 preceding と succeeding の意味は次のとおりである。

preceding	busstopが示すバス停を直前に発車したバスが走行中の区間番号を格納する。busstopが示すバス停からバス停9までの区間に走行中のバスがない場合は9とする。
succeeding	busstopが示すバス停に次に到着するバスが走行中の区間番号を格納する。バス停0からbusstopが示すバス停までの区間に走行中のバスがない場合は−1とする。

バス停 5 をバス B が発車するとき，バス停 5 を直前に発車したバス A が区間 6 を走行し，次にバス停 5 に到着するバス C が区間 3 を走行しているときの例を，図 2 に示す。

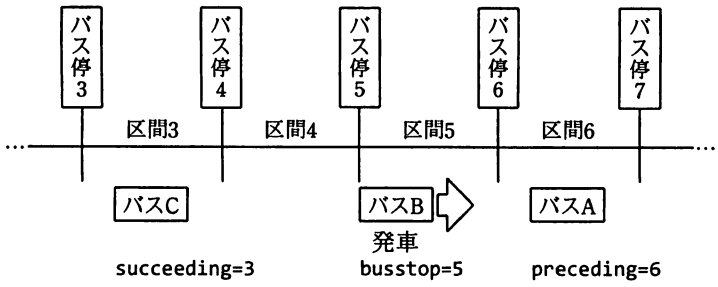

図２　変数 busstop，preceding 及び succeeding の例

(7)　関数 update_wait_time は，バスがバス停 0～8 を発車するとき又はバス停 9 に到着したときに呼び出され，bus_id で識別されるバスの走行位置を更新し，busstop が示すバス停から preceding が示す区間の始まりとなるバス停までの，各バス停に表示する到着待ち時間 route[i].wait_time (i＝busstop, …, preceding) を計算する。

なお，複数台のバスがバス停 0～8 の異なるバス停を同時に発車する場合，又はあるバスがバス停 9 に到着したときに別のバスがバス停 0～8 を発車する場合には，関数 update_wait_time は，バスの一覧 bus の添字の昇順に 1 台分ずつ実行される。

(8)　一つの区間を複数台のバスが同時に走ることはない。

〔プログラム〕

(行番号)

```
 1  #define STPNUM 10        /* バス停の個数 */
 2  #define BUSNUM  5        /* バスの台数 */

 3  typedef struct {
 4      int  std_time;       /* 前のバス停からこのバス停までの
 5                              標準所要時間（分） */
 6      int  wait_time;      /* 到着待ち時間（分） */
 7  } BUSSTOP;

 8  typedef struct {
 9      int  id;             /* バスの車体番号 */
10      int  cur_pos;        /* バスの走行位置 */
11  } BUS;

12  void update_wait_time(int, int, BUSSTOP[STPNUM], BUS[BUSNUM]);

13  void update_wait_time(int bus_id, int busstop,
14                        BUSSTOP route[STPNUM], BUS bus[BUSNUM]) {
15      int preceding = STPNUM - 1, succeeding = -1, i, j;
```

```
16      /* バスの走行位置，succeeding と preceding の更新 */
17      for (j = 0; j < BUSNUM; j++) {
18          if ( [ a ] ) {
19              bus[j].cur_pos = busstop;
20          } else {
21              if (bus[j].cur_pos < busstop) {
22                  if (succeeding < bus[j].cur_pos) {
23                      succeeding = [ b ];
24                  }
25              } else {
26                  if (preceding > bus[j].cur_pos) {
27                      preceding = [ b ];
28                  }
29              }
30          }
31      }

32      /* このバス停の到着待ち時間の計算 */
33      if ( [ c ] ) {
34          route[busstop].wait_time = 0;
35      } else {
36          route[busstop].wait_time = route[busstop].std_time
37                                   + route[busstop - 1].wait_time;
38      }

39      /* 次のバス停から preceding が示すバス停までの到着待ち時間の計算 */
40      for (i = busstop + 1; i <= preceding; i++) {
41          route[i].wait_time = route[i].std_time;
42          if (i != busstop + 1) {
43              route[i].wait_time [ d ];
44          }
45      }
46  }
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　bus_id == bus[j].id　　イ　bus_id != bus[j].id

ウ　bus_id == j　　エ　bus_id != j

bに関する解答群

ア　-1　　イ　busstop　　ウ　bus[j].cur_pos

エ　bus[j].cur_pos + 1　　オ　bus[j].cur_pos - 1　　カ　j

cに関する解答群

ア	`preceding == -1`	イ	`preceding == STPNUM - 1`
ウ	`preceding != -1`	エ	`succeeding == -1`
オ	`succeeding == 0`	カ	`succeeding != 0`

dに関する解答群

ア	`+= route[i - 1].std_time`	イ	`+= route[i - 1].wait_time`
ウ	`+= route[i].std_time`	エ	`= route[i - 1].std_time`
オ	`= route[i - 1].wait_time`	カ	`= route[i].std_time`

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

サービス向上のため，路線上のそれぞれのバス停に取り付けられた表示装置に，到着待ち時間に加えて，そのバスの運行遅延時間を分単位で表示することにした。そのために，関数 update_wait_time に，各バス停に表示する運行遅延時間を計算する処理を追加する。ここで，プログラム中の［ a ］～［ d ］には，正しい答えが入っているものとする。

(1) バスの運行遅延時間は，バス停0から最後に発車したバス停までの標準所要時間の合計と実際にかかった時間の合計の差である。

(2) 関数 update_wait_time の引数に，前のバス停からの実際にかかった時間（分）act_time を追加する。ここで，引数の値に誤りはないものとする。

(3) 構造体 BUSSTOP の要素に，次に到着するバスの運行遅延時間（分）delay_time を追加する。

(4) 構造体 BUS の要素に，バスの運行遅延時間（分）cur_delay を追加する。

(5) バスが定刻よりも早くバス停0～8を発車することはない。

(6) 処理の追加に対応するために，プログラムを表のとおりに変更する。

表　プログラムの変更内容

処置	変更内容

行番号3～14を変更	typedef struct {
	　　int　std_time;　　/* 前のバス停からこのバス停までの
	　　標準所要時間（分）　*/
	　　int　wait_time;　　/* 到着待ち時間（分）　*/
	　　int　delay_time;　　/* 次に到着するバスの運行遅延時間（分）　*/
	} BUSSTOP;
	
	typedef struct {
	　　int　id;　　/* バスの車体番号 */
	　　int　cur_pos;　　/* バスの走行位置 */
	　　int　cur_delay;　　/* バスの運行遅延時間（分）　*/
	} BUS;
	
	void update_wait_time(int, int, int, BUSSTOP[STPNUM], BUS[BUSNUM]);
	
	void update_wait_time(int bus_id, int busstop, int act_time,
	　　BUSSTOP route[STPNUM], BUS bus[BUSNUM]) {

行番号15と16の間に追加	int delay_preceding, delay_succeeding = 0;
行番号19と20の間に追加	bus[j].cur_delay [e]; 　　delay_preceding [f];
行番号23と24の間に追加	delay_succeeding [f];
行番号38と39の間に追加	route[busstop].delay_time = delay_succeeding;
行番号44と45の間に追加	route[i].delay_time = delay_preceding;

eに関する解答群

ア　= act_time - route[busstop].std_time

イ　= bus[j - 1].cur_delay

ウ　= route[busstop].delay_time

エ　+= act_time - route[busstop].std_time

オ　+= bus[j - 1].cur_delay

カ　+= route[busstop].delay_time

fに関する解答群

ア = act_time　　イ = bus[j].cur_delay

ウ = route[busstop].delay_time　　エ += act_time

オ += bus[j].cur_delay　　カ += route[busstop].delay_time